IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of	
Christophe Clavier et al.	Group Art Unit: 2131
Application No.: 09/807,607	Examiner: Kaveh Abrishamkar
Filed: June 1, 2001	Confirmation No.: 2078
For: COUNTERMEASURE METHOD IN AN ELECTRONIC COMPONENT USING A SECRET KEY CRYPTOGRAPHIC ALGORITHM	

Verification of Translation

The undersigned hereby states that:

1. I am fluent in both the French and English languages.

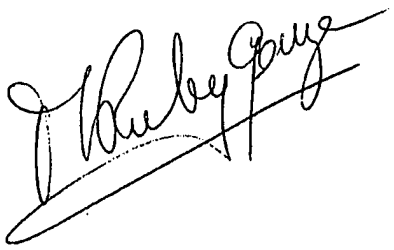
2. I have compared the specification of French patent application No. 98/12989 with the accompanying English-language document.

3. I hereby verify that the accompanying English-language document is a true and correct translation of French patent application No. 98/12989.

Date: 17/02/09

RUBY-GOUZE Michèle

Name:



A COUNTERMEASURE METHOD IN AN ELECTRONIC COMPONENT USING SECRET-KEY CYPTOGRAPHIC ALGORITHM

The present invention relates to a countermeasure method in an electronic component using a secret-key cryptographic algorithm. Such a component is used in applications in which access to services or to data is strictly controlled. It has an architecture formed around a microprocessor and around memories, including a program memory that contains the secret key.

Such components are used in particular in chip cards or "smart cards" for certain applications thereof. For example, such applications are to accessing certain databanks, to remote payment, e.g. for television, to dispensing gasoline, or to going through highway turnpikes or tolls.

Such components or such cards therefore use a secret-key cryptographic algorithm, the best known of which is the Data Encryption Standard (DES) algorithm. Other secret-key algorithms exist, such as the RC5 algorithm or the COMP128 algorithm. Naturally, this list is not exhaustive.

In general and in short, the function of such algorithms is to compute an enciphered message on the basis of a message applied as input (to the card) by a host system (server, cash dispenser or automatic teller, etc.) and on the basis of the secret key contained in the card, and to send back to the host system the enciphered message, thereby making it possible, for example, for the host system to authenticate the component or the card, to interchange data, etc.

Unfortunately, it has become apparent that such components or such cards are vulnerable to attacks that consist in differentially analyzing current consumption and that enable ill-intentioned third parties to find the secret key. Such attacks are referred to as "Differential Power Analysis" ("DPA") attacks.

The principle of such attacks is based on the fact that the power consumption of (i.e. the current consumed by) the microprocessor executing the instructions varies depending on the bit that is manipulated.

In particular, an instruction of the microprocessor that manipulates a data bit generates two different current profiles depending on whether the bit is a "1" bit or a "0" bit. Typically, if the instruction manipulates a "0", then, at that execution instant, a first amplitude is obtained for the consumed current, and if the instruction manipulates a "1", a second amplitude is obtained for the consumed current, which amplitude is different from the first amplitude.

The characteristics of the cryptographic algorithms are well known: computations performed, and

parameters used. The only unknown is the secret key contained in the program memory. The secret key cannot be deduced merely from knowledge of the message applied as input and of the enciphered message sent back.

However, in a cryptographic algorithm, certain items of computed data depend only on the message applied in non-enciphered form to the input of the card, and on the secret key contained in the card. Other items of data computed in the algorithm can also be recomputed only on the basis of the enciphered message (generally delivered in unenciphered form as output from the card to the host system) and of the secret key contained in the card. More precisely, each bit of these particular items of data can be determined on the basis of the input or output message, and of a limited number of particular bits of the key.

Thus, each bit of a particular item of data corresponds to a subkey formed by a particular group of bits of the key.

The bits of such particular items of data that can be predicted are referred to below as "target bits".

The basic idea behind DPA attack is to use the difference in the current consumption profile of an instruction depending on whether it is manipulating a "1" or a "0", and to use the possibility of a target bit being computed by the instructions of the algorithm on the basis of a known input or output message and of a hypothesis on the corresponding subkey.

The principle of DPA attack is thus to test a given subkey hypothesis by applying a selection Boolean function to a large number of current measurement curves, each relating to an input message known to the

attacker, the selection Boolean function being a function of the subkey hypothesis and being defined for each curve by the value predicted for a target bit.

By making a hypothesis on the subkey in question, it is possible to predict the "0" or "1" value that the target bit will take for a given input or output message.

It is then possible for the predicted "0" or "1" value taken by the target bit for the subkey hypothesis in question to be applied as a selection Boolean function for sorting the curves into two packets. A first packet groups together the curves that have seen manipulation of the target bit at "0" and a second packet groups together the curves that have seen manipulation of the target bit at "1" depending on the subkey hypothesis. By averaging the current consumption in each packet, it is possible to obtain a mean consumption curve M0(t) for the first packet and a mean consumption curve M1(t) for the second packet.

If the subkey hypothesis is correct, the first packet really does group together all those curves from among the N curves which have seen the manipulation of the target bit at "0" and the second packet really does group together all those curves from among the N curves which have seen the manipulation of the target bit at "1". The mean consumption curve M0(t) of the first packet then has a mean consumption everywhere except at the moments at which the critical instructions are executed, with a current consumption characteristic of the manipulation of the bit at "0" ($profile_0$). In other words, for all of the curves, all of the manipulated bits have had the same chances of being equal to "0" as

of being equal to "1" except for the target bit which has always taken the value "0". This may be written as follows:

$$M0(t) = [(profile_0 + profile_1)/2]_{t \neq tci} + [profile0]_{tci}$$

i.e.

$$M0(t) = [Vm_t]_{t \neq tci} + [profile_0]_{tci}$$

where tci represents the critical instants at which a critical instruction was executed.

Similarly, the mean consumption curve M1(t) of the second packet corresponds to a mean consumption everywhere except at the moments at which the critical instructions are executed, with a current consumption curve characteristic of the manipulation of the target bit at "1" ($profile_1$). The following can be written:

$$M1(t) = [(profile_0 + profile_1)/2]_{t \neq tci} + [profile1]_{tci}$$

i.e.

$$M1(t) = [Vm_t]_{t \neq tci} + [profile_1]_{tci}$$

As described, the two profiles $profile_0$ and $profile_1$ are not equal. The difference in the curves M0(t) and M1(t) then gives a signal DPA(t) whose amplitude is equal to $profile_0$ - $profile_1$ at the critical instants tci at which the critical instructions manipulating said bit are executed, i.e., in the example shown in Figure 1, at the places tc0 to tc6, and whose amplitude is approximately equal to zero outside the critical instants.

If the subkey hypothesis is incorrect, the sorting does not correspond to reality. Statistically, there are then, in each packet, as many curves having really

seen the manipulation of the target bit at "0" as there are curves having seen the manipulation of the target bit at "1". The resulting mean curve M0(t) is then situated around a mean value given by $(profile_0 + profile_1)/2 = Vm$, because, for each of the curves, all of the manipulated bits, including the target bit have as many chances of taking the value "0" as they have of taking the value "1".

The same reasoning on the second packet leads to a mean current consumption curve M1(t) whose amplitude is situated around a mean value given by $(profile_0 + profile_1)/2 = Vm$

The signal DPA(t) supplied by the difference M0(t) - M1(t) is, in this case, substantially equal to zero. The signal DPA(t) in the case of a incorrect subkey hypothesis is shown in Figure 2.

Thus, the DPA attack makes use of the difference in the current consumption profile while an instruction is being executed, depending on the value of the manipulated bit, to sort current consumption curves on the basis of a Boolean selection function for a given subkey hypothesis. By differentially analyzing the mean power consumption between the two resulting packets of curves, it is possible to obtain an information signal DPA(t).

The procedure of a DPA attack thus consists substantially in the following:

a- drawing N random messages (e.g. N equals 1,000);

b- having the algorithm executed by the card for each of the N random messages, while recording the

current consumption curve each time (as measured on the power supply terminal of the component);

c- making a hypothesis on a subkey;

d- predicting, for each of the random messages; the value taken by one of the target bits whose value depends only on the bits of the (input or output) message and on the subkey taken as a hypothesis, in order to obtain the Boolean selection function;

e- sorting the curves on the basis of this Boolean selection function (i.e. on the basis of the "0" or "1" value predicted for the target bit for each curve under the subkey hypothesis);

f- computing, in each packet, the resulting mean current consumption curve; and

g- determining the difference in said mean curves, so as to obtain the signal DPA(t).

If the hypothesis on the subkey is correct, the Boolean selection function is correct, and the curves of the first packet really do correspond to the curves for which the message applied as input or delivered as output has given a target bit at "0" in the card, and the curves of the second packet really do correspond to the curves for which the message applied as input or delivered as output has given a target bit of "1" in the card.

This is the case shown in Figure 1: the signal DPA(t) is therefore not zero at the instants tc0 to tc6 corresponding to the execution of the critical instructions (i.e. the instructions that manipulate the target bit).

It should be noted that the attacker does not need to know the critical instants with precision. It

suffices for the attacker to have at least one critical instant in the acquisition period.

If the subkey hypothesis is incorrect, the sorting does not correspond to the reality and, in each packet, there are as many curves really corresponding to a target bit at "0" as there are curves corresponding to a target bit at "1". The DPA(t) signal is substantially zero everywhere (case shown in Figure 2). It is necessary to return to step c- and a new subkey hypothesis must be made.

If the hypothesis proves to be correct, it is possible to move on to assessing other subkeys, until the key has been reconstructed to the maximum possible extent. For example, with a DES algorithm, a 64-bit key is used, only 56 bits of which are working bits. With a DPA attack, it is possible to reconstruct at least 48 bits of the 56 working bits.

An object of the present invention is to use a countermeasure method in an electronic component, which method gives a signal DPA(t) that is zero even when the subkey hypothesis is correct.

In this way, nothing makes it possible to distinguish the case when the subkey hypothesis is correct from the case when the subkey hypothesis is incorrect. By means of this countermeasure, the electronic component is protected against DPA attacks.

In the invention, the countermeasure method makes it possible to make the target bits, i.e. the data manipulated by critical instructions, unpredictable.

Because of the countermeasure, for each message applied as input, a target bit manipulated by a critical instruction has an equal probability of taking

the value 0 or the value 1. In each packet of curves that the attacker makes under a given subkey hypothesis, by means of the Boolean selection function that the attacker has computed, there are as many curves that have really manipulated "0" target bits as curves that have really manipulated "1" bits. The signal DPA(t) is always zero, whether the subkey hypothesis is correct or incorrect.

As characterized, the invention therefore provides a countermeasure method in an electronic component using a cryptographic algorithm having a secret key, embodiment of the algorithm comprising the use of first means to deliver output data on the basis of input data, the output data and/or derived data being manipulated by critical instructions. According to the invention, the countermeasure method makes provision to use other means so that the output data and the derived data are unpredictable.

According to the invention, the use of the various means is managed by a one-half probability statistical relationship.

Other characteristics and advantages of the invention are described in more detail in the following description given by way of non-limiting example and with reference to the accompanying drawings, in which:

Figures 1 and 2 (described above) show the DPA(t) signal that can be obtained as a function of a hypothesis on a subkey of the secret key K, in a DPA attack;

Figures 3 and 4 are flow charts showing how the first rounds and the last rounds of the DES algorithm are executed;

Figure 5 is a block diagram of the SBOX operation used in the DES algorithm;

Figure 6 shows an example of a one-input and one-output elementary table of constants used in the SBOX operation;

Figures 7 and 8 show examples of flow charts of execution of the first and last rounds of the DES algorithm, in an embodiment of the countermeasure method of the invention;

Figures 9 and 10 respectively show second and third elementary tables of constants of the invention;

Figure 11 is a general flow chart of the execution of the DES in an embodiment of the countermeasure method of the invention; and

Figure 12 is a simplified block diagram of a smart card including an electronic component in which the countermeasure method of the invention is useed.

The present invention is explained below in an example of application to the DES cryptographic algorithm. The invention is not limited to this example alone. It is applicable to secret-key cryptographic algorithms in general.

The DES cryptographic algorithm (referred to below as the "DES" or as the "DES algorithm") comprises 16 computation rounds, referenced T1 to T16, as shown in Figures 3 and 4.

The DES begins with an initial permutation IP on the input message M (Figure 3). The input message M is a 64-bit word f. After permutation, a 64-bit word e is obtained that is cut in two to form the input parameters L0 and R0 of the first round (T1). L0 is a word d having 32 bits and containing the 32 most

significant bits of the word e. R0 is a word h having 32 bits and containing the 32 least significant bits of the word e.

The secret key K, which is a 64-bit word g, itself undergoes permutation and compression to deliver a word r having 56 bits.

The first round comprises an operation EXP PERM on the parameter R0, which operation consists in expansion and permutation so as to deliver as output a word l having 48 bits.

This word l is combined with a parameter K1 in an EXCLUSIVE OR type operation referenced XOR so as to deliver a word b having 48 bits. The parameter K1 which is a 48-bit word m is obtained from the word r by shifting by one position (operation referenced SHIFT in Figures 3 and 4) followed by permutation and by compression (operation referenced COMP PERM).

The word b is applied to an operation referenced SBOX at the output of which a word a having 32 bits is obtained. This particular operation is explained in more detail with reference to Figures 5 and 6.

The word a undergoes a permutation P PERM, giving as output the 32-bit word c.

This word c is combined with the input parameter L0 of the first round T1, in a logic EXCLUSIVE OR type operation, referenced XOR, which delivers as output the 32-bit word g.

The word h (=R0) from the first round supplies the input parameter L1 of the following round (T2), and the word g from the first round supplies the input parameter R1 of the following round. The word p from

the first round supplies the input $\underline{r}$ of the following round.

The other rounds T2 to T16 proceed similarly except as regards the SHIFT operation which is performed on one or two positions depending on the round in question.

Each round T1 thus receives as input the parameters Li-1, Ri-1, and $\underline{r}$ and delivers as output the parameters Li, Ri, and $\underline{r}$ for the following round Ti+1.

At the end of the DES algorithm (Figure 4), the enciphered message is computed on the basis of the parameters L16 and R16 delivered by the last round T16.

The enciphered message C comprises, in practice, the following operations:

- forming a 64-bit word e' by inverting the positions of the words L16 and R16, and then by concatenating them; and

- applying the permutation IP^{-1} that is the inverse of the permutation at the beginning of the DES, so as to obtain the 64-bit word f' forming the enciphered message C.

The SBOX operation is shown in detail in Figures 5 and 6. It comprises a constants table TC_0 for delivering output data $\underline{a}$ as a function of input data $\underline{b}$.

In practice, this constants table TC_0 is in the form of eight elementary constants tables TC_01 to TC_08, each receiving as input only 6 bits of the word $\underline{b}$, so as to deliver as output only 4 bits of the word $\underline{a}$.

Thus, the elementary constants table TC_01 shown in Figure 6 receives as input data the bits b1 to b6 of the word $\underline{b}$ and delivers as output the bits a1 to a4 of the word $\underline{a}$.

In practice, these eight elementary constants tables are stored in a program memory of the electronic component.

In the SBOX operation of the first round T1, a particular bit of the output data a of the constants table TC_0 depends on only 6 bits of the data b applied as input, i.e. on only 6 bits of the secret K and of the input message (M).

In the SBOX operation of the last round T16, a particular bit of the output data a of the constants table TC_0 can be recomputed on the basis of only 6 bits of the secret key K and of the enciphered message (C).

Going back to the principle of the DPA attack, if the target bit is chosen to be a bit of the output data a, it suffices to make a hypothesis on 6 bits of the key K in order to predict the value of a target bit for a given input message (M) or for a given output message (C). In other words, for the DES, it suffices to make a hypothesis on a 6-bit subkey.

In a DPA attack on such an algorithm for a given target bit, it is necessary to distinguish a correct subkey hypothesis from 64 possible hypothesis.

Thus, by taking only eight bits from the word a as target bits (one output bit per elementary constants table TC_01 to TC_08), it is possible to discover up to $6 \times 8 = 48$ bits of the secret key by making DPA attacks on each of the target bits.

In the DES, instructions that are critical for DPA attacks are to be found at the beginning of the algorithm and at the end of the algorithm.

At the beginning of the DES algorithm, the items of data that can be predicted on the basis of an input

message M and of a subkey hypothesis are the items of data a and g computed in the first round (T1).

The data a of the first round T1 (Figure 3) is the output data of the SBOX operation of the round in question. The data g is computed on the basis of the data a, by permutation (P PERM) and EXCLUSIVE OR operation with the input parameter L0.

The data c of the first round is data derived from the data a of the first round. The derived data c corresponds to a single permutation of bits of the data a.

The data l of the second round is data derived from the data g of the first round, because it corresponds to a permutation of the bits of the word g, certain bits of the word g further being duplicated.

Knowing a and g, it is possible also to know said derived data.

The critical instructions of the beginning of the algorithm are the critical instructions that manipulate either the data that can be predicted or the data a of the first round, or else derived data.

The critical instructions manipulating the data a of the first round T1 or the derived data c are thus instructions of the end of the SBOX operation, of the P PERM operation, and of the beginning of the XOR operation of the first round T1.

The critical instructions manipulating the data g or derived data are all the instructions from the end of the XOR operation of the first round T1 to the instructions of the start of the SBOX operation of the second round T2, and the instructions of the start of

the XOR operation of the end of the end of the third round T3 (L2 = h(T2) = g(T1)).

At the end of the DES algorithm, the items of data that can be predicted on the basis of an enciphered message C and of a subkey hypothesis are the data a of the sixteenth round T16 and the data L15 equal to the word h of the fourteenth round T14.

The critical instructions manipulating the data a of the sixteenth round or derived data are the instructions of the sixteenth round for the end of the SBOX operation, for the P PERM permutation operation and for the start of the XOR operation.

For the data L15, the critical instructions manipulating said data or the derived data are all the instructions from the end-of-XOR-operation instructions of the fourteenth round T14 to the start-of-SBOX-operation instructions of the fifteenth round T15, and the start-of-XOR-operation instructions of the end the sixteenth round T16.

The countermeasure method of the invention as applied to this DES algorithm consists in having, for each critical instruction, as many chances that the critical instruction is manipulating an item of data as that it is manipulating the complement of said item of data. Thus, regardless of the target bit on which the DPA attack can be made, there are as many chances that the critical instructions that manipulate the bit are manipulating a "1" as that they are manipulating a "0".

In practice, this must apply for each of the potential target bits: in other words, since the attacker has the choice between a plurality of possible attacks, i.e. between a plurality of possible Boolean

selection functions for performing the curve sorting, for any given subkey hypothesis, the embodiment of the countermeasure method of the invention must make sure that the data manipulated by each of the critical instructions takes randomly, and every other time, a value or the complement of that value. As regards applying the countermeasure method of the invention to the DES algorithm, it is thus necessary to apply the countermeasure to the critical instructions at the beginning of the DES, and to the critical instructions at the end of the DES, in order to be fully protected.

In the DES, all of the data manipulated by critical instructions is output data or data derived from output data of an SBOX operation.

At the beginning of the DES, the data that can be predicted is the data $\underline{a}$ and $\underline{g}$ of the first round T1. The data $\underline{a}$ is the output data of the SBOX operation of the first round. The data $\underline{g}$ is computed on the basis of the data $\underline{a}$, since $\underline{g}$ = P PERM(a) XOR L0. $\underline{g}$ is thus data derived from the output data $\underline{a}$ of the SBOX operation of the first round. Thus, all of the data manipulated by the critical instructions of the beginning of the DES result directly or indirectly from the output data $\underline{a}$ of the SBOX operation of the first round.

As regards the end of the DES, the data that can be predicted is the data $\underline{a}$ of the sixteenth round T16 and the data $\underline{g}$ of the fourteenth round T14, $\underline{g}$ being equal to L15.

The data $\underline{a}$ is the output data of the SBOX operation of the sixteenth round T16.

As regards the data L15, it is computed, during normal execution of the DES algorithm, on the basis of the output data a of the SBOX operation of the fourteenth round T14: L15 = P PERM (a) XOR L14.

If the items of output data a of the particular SBOX operations are made unpredictable, all of the items of derived data are also made unpredictable: thus all of the items of data manipulated by the critical instructions of the DES algorithm are made unpredictable.

The SBOX operation thus corresponds to first means which consist of a constants table TC_0, and which are used in each round to deliver output data E on the basis of input data S.

An embodiment of the countermeasure method as applied to the DES algorithm may consist in using at least one other constants table as other means for making the output data a unpredictable, so that all of the items of said output data and/or of derived data manipulated by the critical instructions are unpredictable.

In executing the algorithm, the use of the various means, i.e., in the example, of the various constants tables is managed on the basis of a one-half probability statistical relationship.

The other constants table or the other constants tables are such that they make the complemented item of data correspond to one and/or the other of the items of input data d and of output data of the first constants table TC_0.

Figures 7 and 8 thus show an embodiment of the countermeasure of the invention as applied to the DES algorithm.

Figure 7 shows the beginning of the algorithm. The data operations that are not modified by the countermeasure method of the invention bear the same references as in Figure 3 described above.

At the beginning of the DES algorithm, a second constants table TC_1 is provided in the SBOX operation of the first round T1. All of the items of data affected by the second constants table TC_1 are given a ' sign or a $\bar{}$ sign in these figures. It can be seen that the critical instructions of the beginning of the DES manipulate all of the items of data affected by the countermeasure method.

It can be observed that, since the first constants table is made up of eight first constants tables, the second constants table is also made up of eight second constants tables.

In the emdobiment shown, the first constants table TC_0 and the second constants table TC_1 are such that, for the same input data E, the second table delivers as output the complement $\bar{S}$ of the output data S delivered by the first table.

Figure 9 shows such a second elementary table TC_11 delivering output that is complemented relative to the first elementary table TC_01 shown in Figure 6.

With such a second constants table TC_1, the complement $\bar{a}$ of the data $\underline{a}$ obtained with the first constants table TC_0 is obtained at the output the SBOX operation of the first round T1. Similarly, in the

first round T1, the complemented data $\bar{g}$ is obtained, and, in the second round T2, the complemented items of data $\bar{h}$, $\bar{L2}$, $\bar{l}$, and $\bar{b}$ are obtained.

By using the first table or the second table to deliver the output data on the basis of a one-half probability statistical relationship, all of the potential target bits at the start of the DES that are manipulated by the critical instructions have as many chances of taking the value "1" as they do of taking the value "0".

At the end of the DES algorithm, using the countermeasure method of the invention requires the use of a plurality of constants tables that are different from the first table because it is necessary to consider both the output data $\underline{a}$ computed in the fourteenth round T14, and the output data $\underline{a}$ computed in the sixteenth round T16 in order to make all of the items of data manipulated by the critical instructions of the end of the DES unpredictable.

An embodiment of the countermeasure method as applied to the end of the DES algorithm is shown in Figure 8.

It makes provision to use two constants tables TC_1 and TC_2.

In the SBOX operation of the fourteenth round T14, the second constants table TC_1 (already used for the beginning of the DES) is used.

And a third constants table TC_2 is used in the SBOX operations of the fifteenth and sixteenth rounds.

This third constants table TC_2 is such that it supplies the complement $\bar{S}$ of the output data S to the

complement $\bar{E}$ of the input data E of the first constants table TC_0. An example of a corresponding third elementary constants table TC_21, on the basis of the first elementary constants table TC_0, is shown in Figure 10.

By using such constants tables, it appears in Figure 8 that all of the critical instructions manipulate complemented data.

The invention is not limited to these examples of constants tables TC_1 and TC_2. Other possibilities exist. For example, for the countermeasure method applied at the end of the DES, it is also possible to combine use of the constants table TC_1 with use of another constants table defined relative to the first constants table TC_0 as supplying the output data S to the complement $\bar{E}$ of the input data.

In general, the end of the DES requires the use of different constants tables, as a function of the rounds in question, so that all of the items of data manipulated by the critical instructions of the end of the DES are unpredictable.

The embodiment described with reference to Figures 7 and 8 does have a drawback however: the countermeasure method applied at the input of the DES produces intermediate computed results L3' and R3' which are incorrect. All of the following intermediate results are therefore incorrect as well.

Similarly, at the end of the DES, the countermeasure method applied at the input of the DES produces intermediate computed results L16' and R16' that are incorrect.

In all cases, the enciphered message is incorrect.

In this embodiment of the invention, it is thus necessary to make provision to be capable of resuming the remainder of the algorithm with the correct intermediate results every time, once the critical instructions have been passed.

In practice, since the critical instructions of the start of the DES are to be found in the first three rounds, these three rounds are duplicated. In other words, provision is made to execute two sequences, each of which comprises at least the first three rounds T1, T2, T3. A first sequence SEQA uses the first constants table TC_0 in each round. The other sequence SEQB uses the second constants table TC_1 at least in the first round T1. In the example shown, the first constants table is used in the two following rounds T2 an T3.

As described, in the countermeasure method of the invention, the use of the various means, i.e. in the example, the use of the various constants tables, is managed on the basis of a one-half probability statistical relationship. This one-half probability statistical relationship is then more particularly applied to the order in which the various means are used, i.e., in the example, to the order in which the two sequences SEQA and SEQB are executed.

Similarly, to have the correct parameters L16 and R16 at the end of the DES for forming the enciphered message C, the three rounds T14, T15, and T16 (Figure 7) that contain the end-of-DES critical instructions are also duplicated. Two sequences are thus executed, each of which comprises at least the last three rounds T14, T15, T16. A first sequence SEQA' uses the first

constants table TC_0 in each round. The other sequence SEQB' uses the other constants tables TC_1 and TC_2. As above, the one-half probability statistical relationship is then applied to the order in which the two sequences SEQA' and SEQB' are executed.

The critical instructions are then executed twice, once in each sequence. But, at the time at which any of the critical instructions of either one of the sequences is executed, the probability of manipulating an item of data is equal to the probability of manipulating its complement.

The computation program of the DES that is useed in the electronic component must therefore be modified to include the countermeasure method of the invention. An example of an execution flow chart of the invention, using the countermeasure method at the start of and at the end of the DES, as in the embodiment described with reference to Figures 7 and 8 is shown in Figure 11. In this example, each of the sequences SEQA and SEQB comprise the first three rounds and each of the sequences SEQA' and SEQB' comprise the last three rounds.

The computation program then consists mainly, at the beginning of computation, in saving the input parameters referenced DATAIN and KEY which, in practice, correspond to the parameters L0, R0, and $\underline{r}$, in a temporary memory zone referenced CONTEXT0.

In this computation program, a first loop counter FR is then positioned at 0, and a value RND1 equal to 0 or to 1 is drawn randomly.

In the example, if RND1 is equal to 1, the sequence SEQB of T1, T2, T3 is performed first, in

which sequence the second constants table TC_1 is used in round T1, and the first table TC_0 is used for rounds T2 and T3. The output parameters L3', R3' (which have incorrect values) are saved in a temporary memory zone referenced CONTEXT2.

If FR is not equal to 1, it is put to 1, the input parameters are retrieved from CONTEXT0, and the value of RND1 is complemented. In the example, RND1=0 is obtained. The other sequence SEQA of T1, T2, T3 is then executed in which the first constants table is used in all three rounds T1, T2, and T3. The output parameters (correct values) are saved in a temporary memory zone referenced CONTEXT1.

If FR is at 1, both sequences have been performed. CONTEXT1 is then retrieved to deliver the intermediate results L3, R3 having the correct values, in the next round (T4).

If RND1 is equal to zero, the procedure begins with $T1(TC_0)$, $T2(TC_0)$, $T3(TC_0)$ and finishes with $T1(TC_1)$, $T2(TC_0)$, $T3(TC_0)$.

At the end of the round T13, the parameters L13 and R13 delivered by this round are saved in the temporary memory CONTEXT0, and the procedure continues for the remaining rounds T14, T15, and T16 in a manner similar to the first rounds.

In all cases, the number of instructions must be exactly the same regardless of the computation path. This is why, in particular, in the application example described, provisions is made also to save the incorrect values (L3', R3', or L16', R16') in the temporary memory zone CONTEXT2.

If any difference exits between the two possible paths, there is then a possibility of fruitful DPA attack.

The countermeasure method of the invention is not limited to the particular embodiment described with reference to the algorithm DES. It is applicable to any secret-key cryptographic algorithm. In general, for any embodiment of an algorithm comprising the use of first means for delivering output data on the basis of input data, with the output data and/or derived data being manipulated by critical instructions, the countermeasure method of the invention comprises the use of other means, so that the output data and the derived data are unpredictable.

The use of the various means, i.e. of the first means and of the other means, is managed using a one-half probability statistical relationship.

The other means may comprise a plurality of different means. They are such that they make the complemented item of data correspond to one or other of the items of input data or of output data of the first means.

In the example described wherein the countermeasure method is applied to DES, the first means consist of the first constants table TC_0. The other means consist, at the start of the DES, of the second constants table TC_1. At the end of the DES, they consist of two different constants tables TC_1 and TC_2, in the example.

In order to apply the countermeasure method of the invention to a given secret-key cryptographic algorithm, it is necessary firstly to determine all

those items of data of this algorithm which can be predicted, and all those critical instructions (i.e. instructions that are critical in the DPA attack sense) which manipulate such items of data or derived items of data. It is then necessary to identify in the algorithm first means and other means in the sense of the invention, so that all of the data manipulated by the critical instructions is unpredictable. For the DES algorithm, the first means are the constants table TC_0. The other means are, for example, other constants tables. These means may be different operations for other algorithms. For the same algorithm, the means may consist of different operations depending on the identified critical instructions.

The electronic component 1 using such a countermeasure method in a secret-key cryptographic algorithm typically comprises, as shown in Figure 12, a microprocessor μP, a program memory 2, and a working memory 3. To be capable of managing the use of the various means of the invention, which are, in the example described, the various constants tables stored in the program memory, means 4 for generating a random 0 or 1 value are provided that, with reference to Figure 11, deliver the value of RND1 each time the DES is executed. Such a component may particularly be used in a smart card CP to improve its tamper-resistance.

CLAIMS

1/ A countermeasure method in an electronic component using a cryptographic algorithm having a secret key (K), use of the algorithm comprising the use of first means (TC_0) to deliver output data (S) on the basis of input data (E), the output data and/or derived data being manipulated by critical instructions, said countermeasure method being characterized in that it makes provision to use other means (TC_1) so that the output data and the derived data are unpredictable.

2/ A countermeasure method according to claim 1, characterized in that the use of the various means (TC_0, TC_1) is managed by a one-half probability statistical relationship.

3/ A countermeasure method according to claim 2, the use of the algorithm comprising sixteen computation rounds (T1, ..., T16), said countermeasure method being characterized in that it comprises executing a first sequence (SEQA) and a second sequence (SEQB), each of which is made up of at least the first three rounds (T1, T2, T3), the order in which the sequences are executed being a function of the one-half probability statistical relationship, the first sequence (SEQA) using the first means (TC_0) in each round, and the second sequence (SEQB) using the other means (TC_1) in at least the first round (T1).

4/ A countermeasure method according to claim 3, characterized in that each of the first and second sequences is made up of the first three rounds (T1, T2, T3).

5/ A countermeasure method according to claim 3 or 4, characterized in that the other means consist of second means (TC_1) such that, for the same input data (E), they deliver as output the complement ($\bar{S}$) of the output data (S) of the first means (TC_0).

6/ A countermeasure method according to claim 2, the use of the algorithm comprising sixteen computation rounds (T1, ..., T16), said countermeasure method being characterized in that it comprises executing a first sequence (SEQA') and a second sequence (SEQB'), each of which is made up of at least the last three rounds (T14, T15, T16), the order in which the sequences are executed being a function of the one-half probability statistical relationship, the first sequence (SEQA') using the first means (TC_0) in each round, the second sequence (SEQB') using the other means (TC_1, TC_2).

7/ A countermeasure method according to claim 6, characterized in that each of the first and second sequences is made up of the last three rounds, and in that the other means used in the second sequence comprise second means (TC_1) and third means (TC_2).

8/ A countermeasure method according to claim 6 or 7, characterized in that the second means (TC_1) are such that, for the same input data (E), they deliver as output the complement ($\bar{S}$) of the output data (S) of the first means (TC_0), and in that the second means are used in the second sequence (SEQB') for the fourteenth round (T14).

9/ A countermeasure method according to claim 8, characterized in that the third means (TC_2) are such that, for the complement of the input data (E), they

deliver as output the complement ($\bar{S}$) of the output data (S) of the first means (TC_0) and they are used in the second sequence for the fifteenth round and the sixteenth round (T15, T16).

10/ A countermeasure method according to any preceding claim, characterized in that the various means are constants tables.

11/ An electronic component using the countermeasure method according to any preceding claim, said electronic component being characterized in that the various means (TC_0, TC_1, TC_2) for delivering output data on the basis of input data are fixed in a program memory of said component, and in that said component includes means for generating a random 0 or 1 value (RND1) for managing the use of said various means.

12/ A smart card including an electronic component according to claim 11.

ABSTRACT

In an electronic component using a cryptographic algorithm having a secret key (K), use of the algorithm comprising the use of first means (TC_0) to deliver output data (S) on the basis of input data (E), the output data and/or derived data being manipulated by critical instructions, a countermeasure method makes provision to use other means (TC_1 and/or TC_2) so that the output data and the derived data are unpredictable.

FIG.1

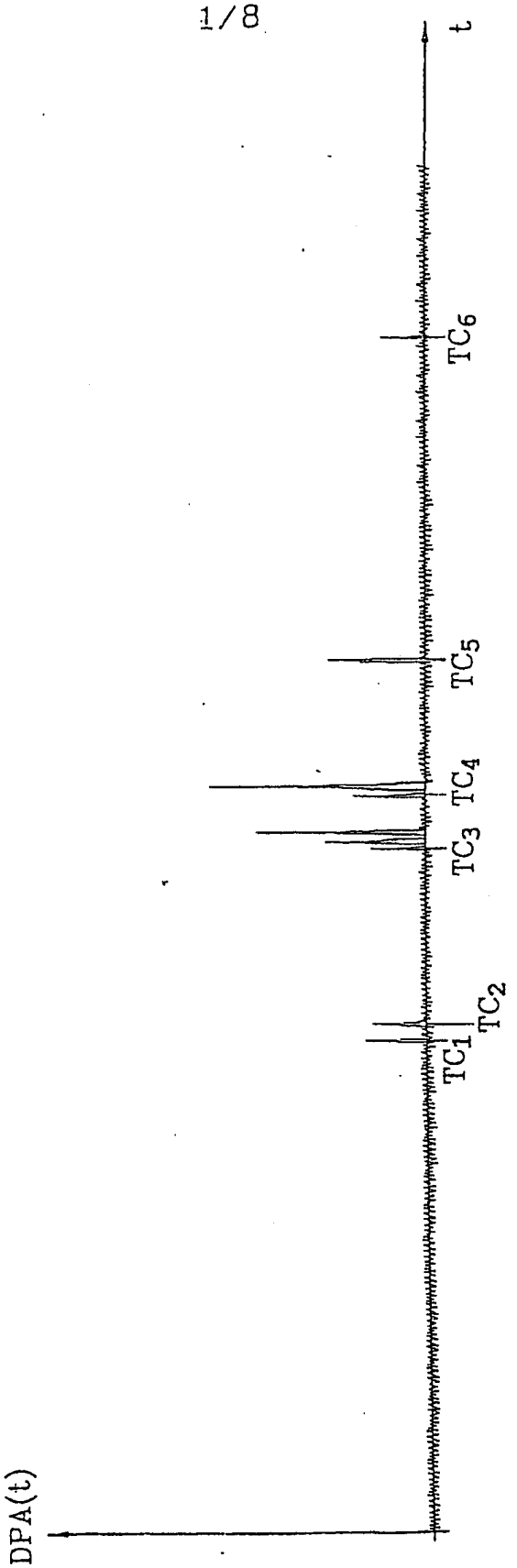
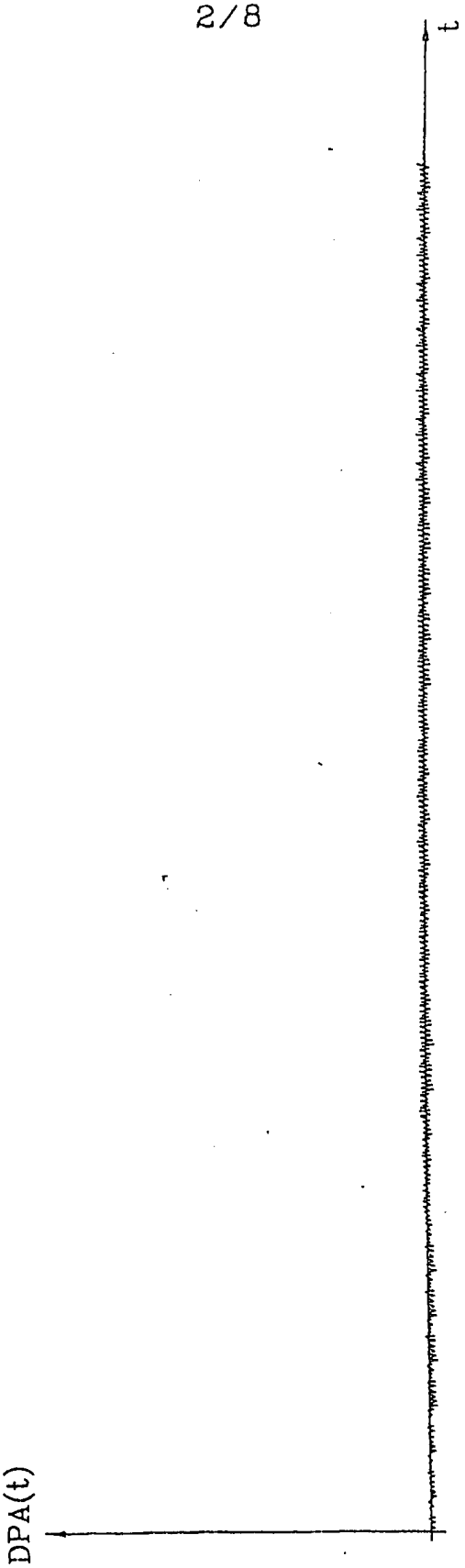


FIG.2



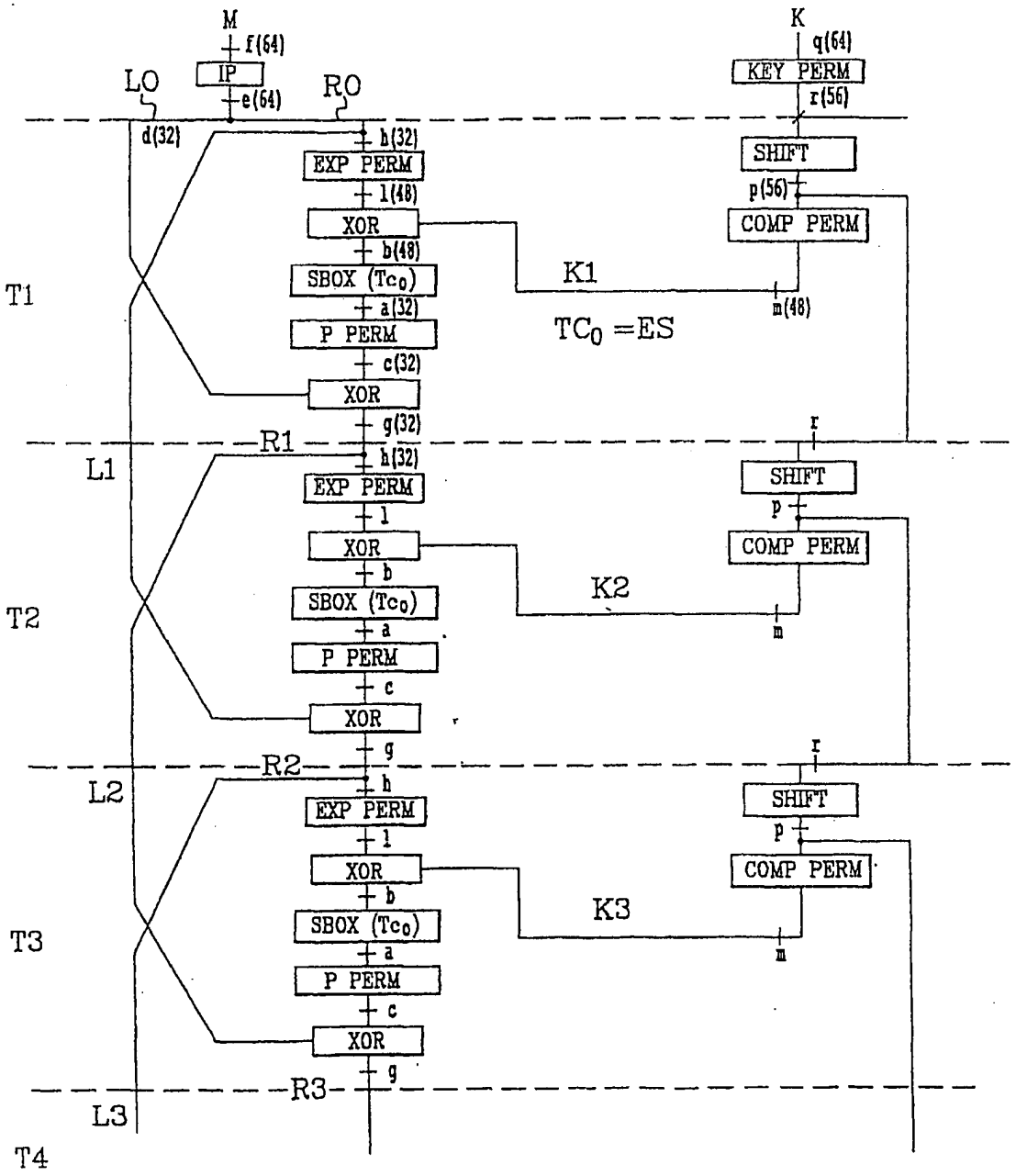

M
f(64)
IP
e(64)
L0
R0
d(32)
b(32)
EXP PERM
l(48)
XOR
b(48)
SBOX (Tc0)
a(32)
P PERM
c(32)
XOR
g(32)
R1
L1
T1
K
q(64)
KEY PERM
r(56)
SHIFT
p(56)
COMP PERM
K1
m(48)
TC0 =ES
r
SHIFT
p
COMP PERM
K2
m
T2
R2
L2
T3
K3
R3
L3
T4

FIG.3

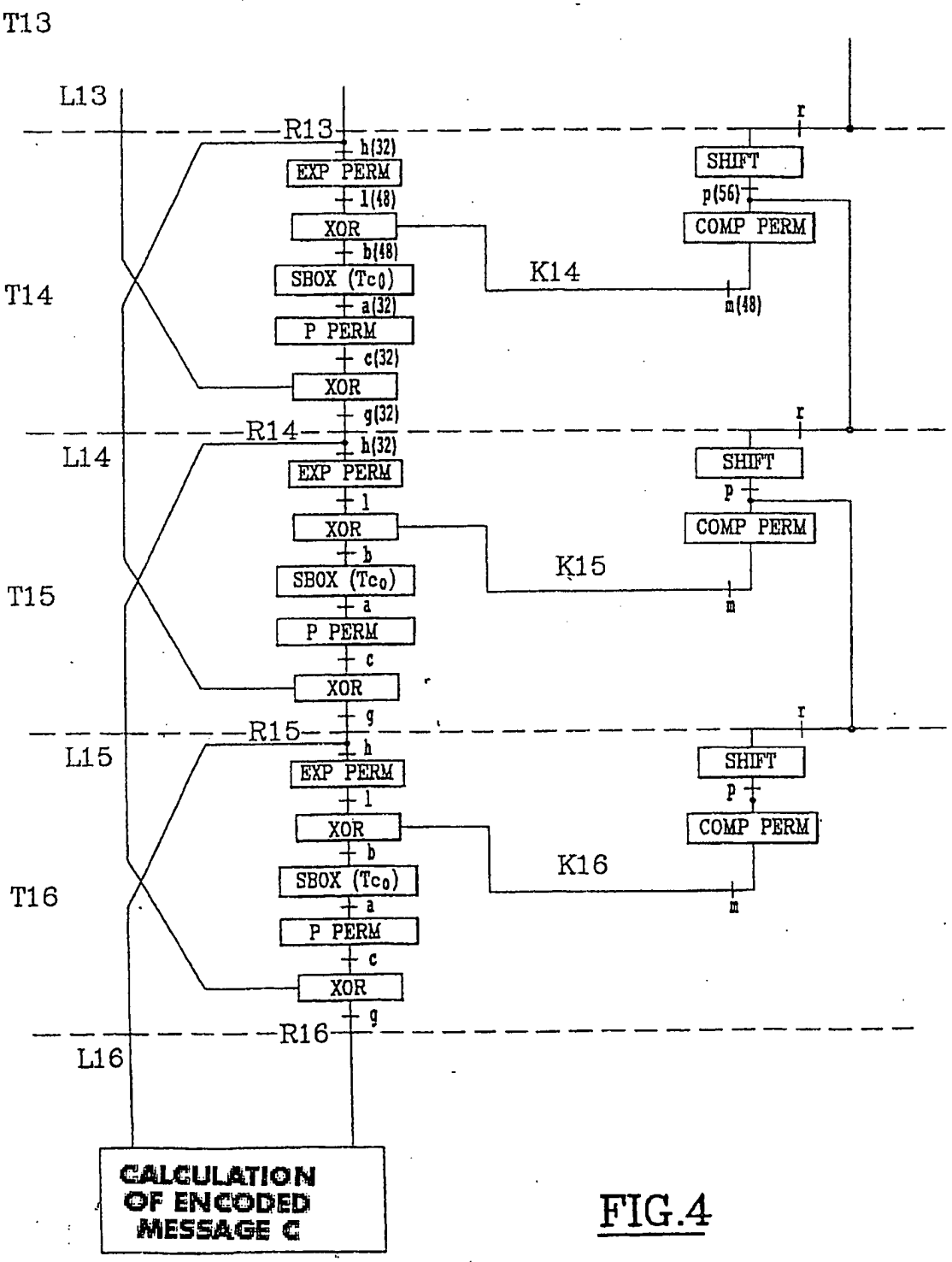

T13
L13
R13
h(32)
EXP PERM
l(48)
XOR
b(48)
SBOX (Tc0)
a(32)
P PERM
c(32)
XOR
g(32)
T14
R14
L14
h(32)
EXP PERM
l
XOR
b
SBOX (Tc0)
a
P PERM
c
XOR
g
T15
R15
L15
h
EXP PERM
l
XOR
b
SBOX (Tc0)
a
P PERM
c
XOR
g
T16
R16
L16
CALCULATION OF ENCODED MESSAGE C
r
SHIFT
p(56)
COMP PERM
K14
m(48)
r
SHIFT
p
COMP PERM
K15
m
r
SHIFT
p
COMP PERM
K16
m

FIG.4

FIG.5

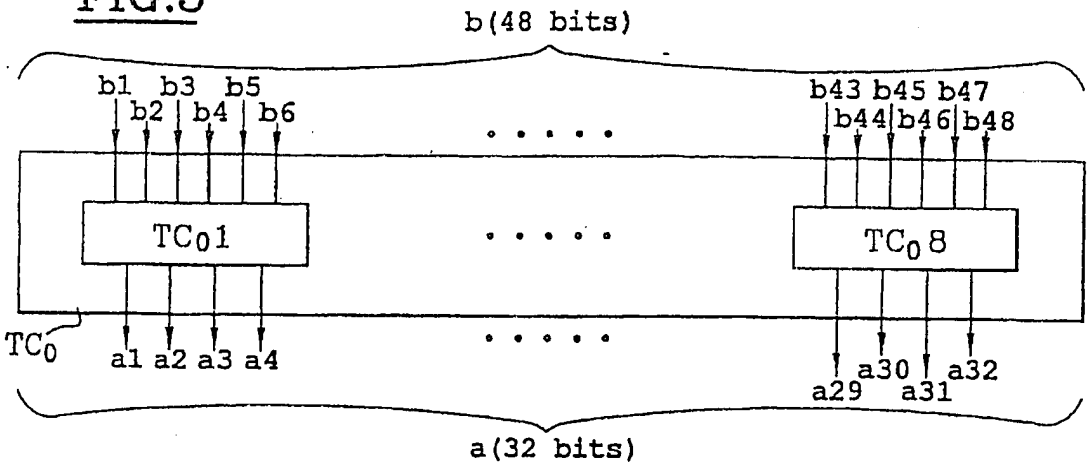


FIG.6

TC0 1

E1=b1b2b3b4b5b6	S1=a1a2a3a4
000000	1101
000001	0101
⋮	⋮
111111	1010

FIG.9

TC1 1

E1=b1b2b3b4b5b6	/S1=$\overline{a1a2a3a4}$
000000	0010
000001	1010
⋮	⋮
111111	0101

FIG.10

TC2 1

/E1=$\overline{b1b2b3b4b5b6}$	/S1=$\overline{a1a2a3a4}$
000000	0101
⋮	⋮
111110	1010
111111	0010

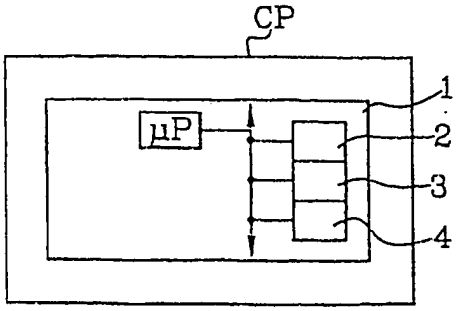


FIG.12

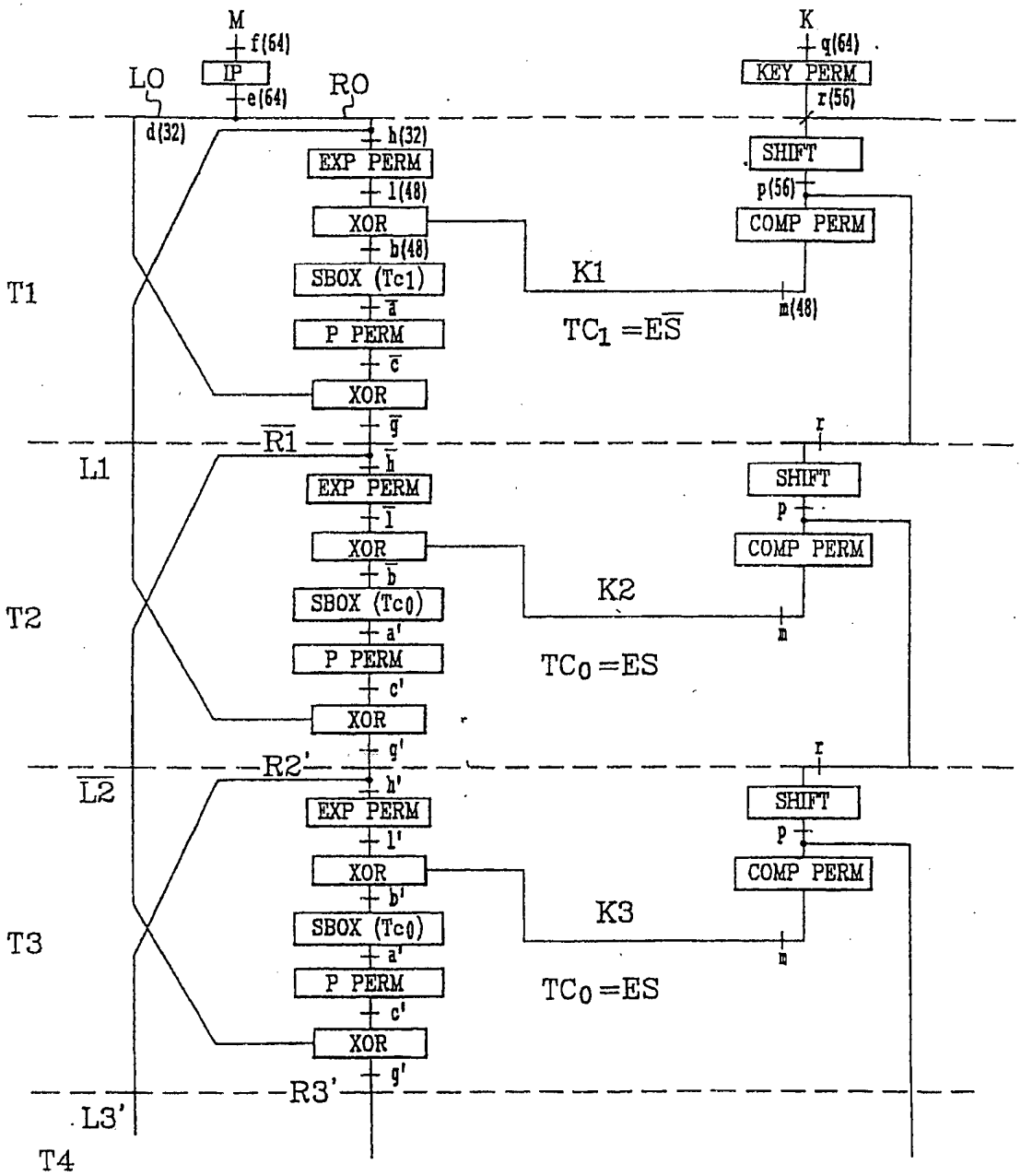

M
f(64)
IP
e(64)
LO
RO
d(32)
h(32)
EXP PERM
l(48)
XOR
b(48)
SBOX (Tc1)
a
P PERM
c
XOR
g
R1
L1
T1
K
q(64)
KEY PERM
r(56)
SHIFT
p(56)
COMP PERM
K1
m(48)
TC1 =ES
r
SHIFT
p
COMP PERM
K2
m
TC0 =ES
T2
h
EXP PERM
l
XOR
b
SBOX (Tc0)
a'
P PERM
c'
XOR
g'
R2'
L2
r
SHIFT
p
COMP PERM
K3
m
TC0 =ES
T3
h'
EXP PERM
l'
XOR
b'
SBOX (Tc0)
a'
P PERM
c'
XOR
g'
R3'
L3'
T4

FIG.7

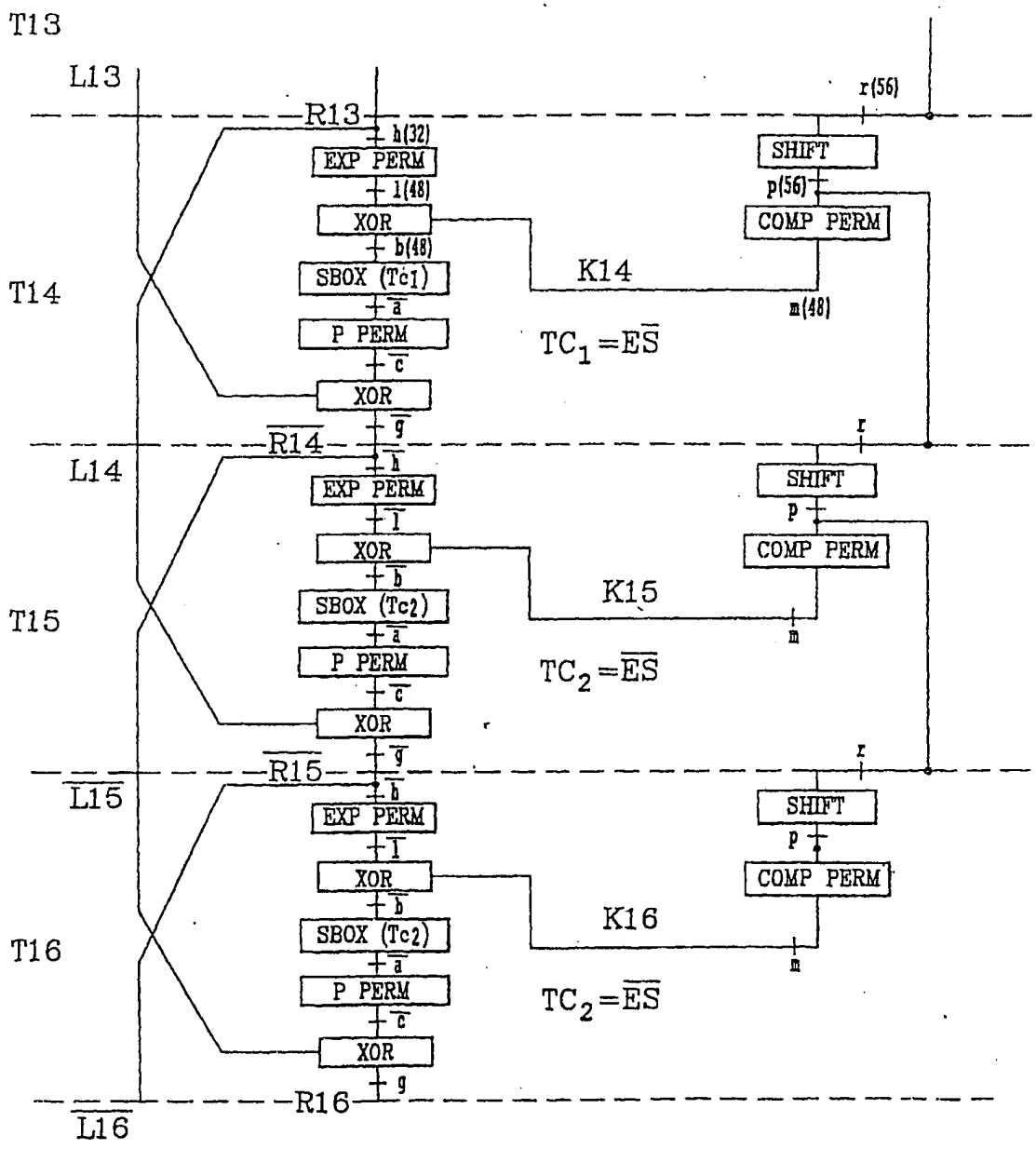

T13
L13
R13
h(32)
EXP PERM
l(48)
XOR
b(48)
SBOX (Tc1)
a
P PERM
c
XOR
g
r(56)
SHIFT
p(56)
COMP PERM
K14
m(48)
TC1=ES
T14
R14
L14
h
EXP PERM
l
XOR
b
SBOX (Tc2)
a
P PERM
c
XOR
g
r
SHIFT
p
COMP PERM
K15
m
TC2=ES
T15
R15
L15
h
EXP PERM
l
XOR
b
SBOX (Tc2)
a
P PERM
c
XOR
g
r
SHIFT
p
COMP PERM
K16
m
TC2=ES
T16
R16
L16

FIG.8

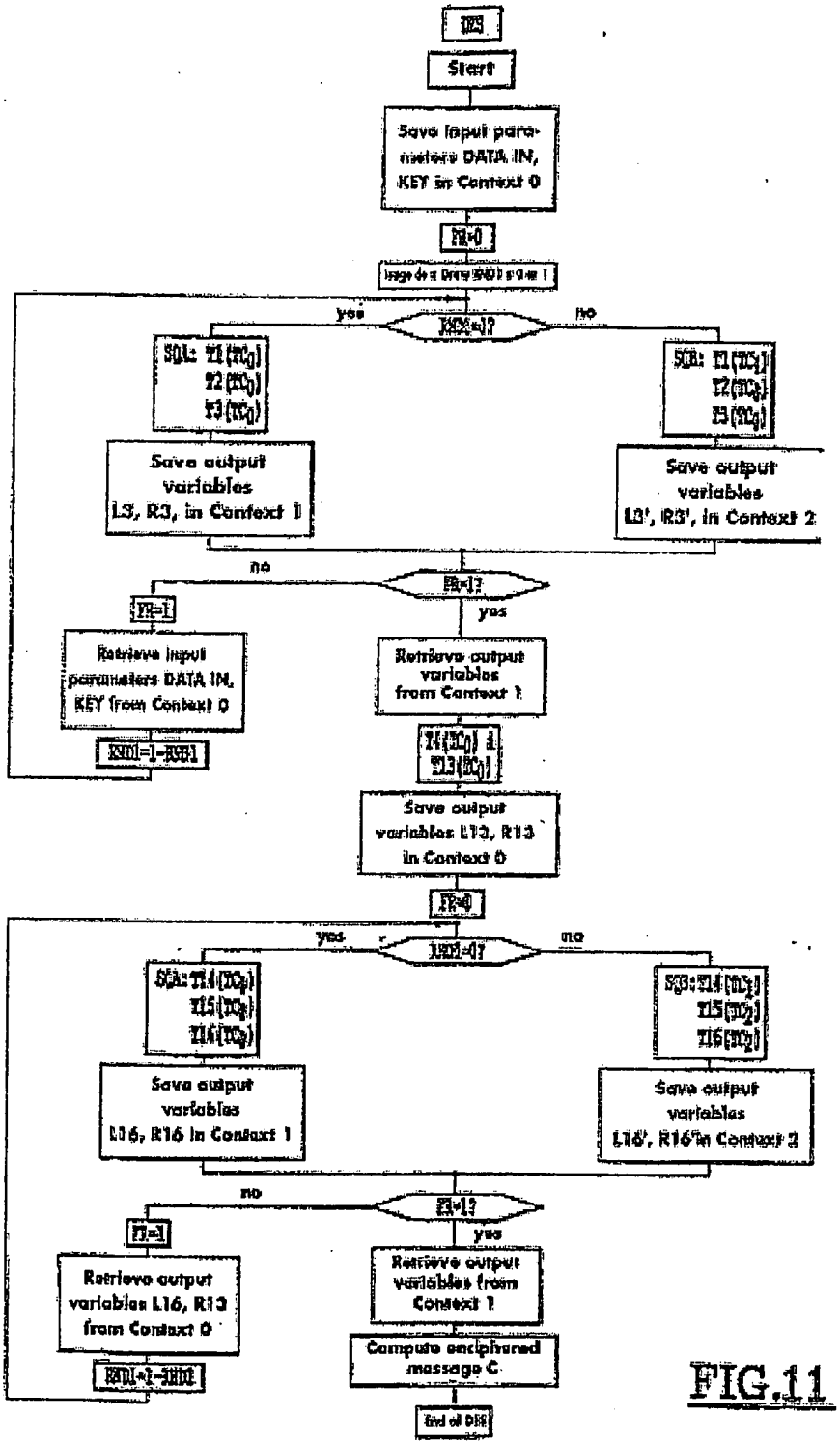

Start
Save input parameters DATA IN, KEY in Context 0
yes
no
Save output variables L3, R3, in Context 1
Save output variables L3', R3', in Context 2
no
yes
Retrieve input parameters DATA IN, KEY from Context 0
Retrieve output variables from Context 1
Save output variables L16, R16 in Context 0
yes
no
Save output variables L16, R16 in Context 1
Save output variables L16', R16' in Context 2
no
yes
Retrieve output variables L16, R16 from Context 0
Retrieve output variables from Context 1
Compute enciphered message C

FIG.11